Improving Large Language Models With Combinatorial Optimization

Mert Esencan*, Tarun Kumar, Can Unlu, Alan Ho

Introduction: The rise in popularity of autoregressive architectures for artificial intelligence such as Large Language Models (LLMs) marks a significant step towards artificial general intelligence (AGI). However, the same architecture makes these LLMs prone to hallucination—defined as generation that is nonsensical or unfaithful to source material [1]. Furthermore, it is commonly accepted that LLMs lack the ability to plan or reason, which limits their potential in high-value tasks requiring those skills.

One approach to handling these limitations is to provide additional retrieved context to reduce improper generations. Techniques such as retrieval augmented generation (RAG) query a vector database to retrieve source material before generating the LLM response [2]. This approach is particularly suitable for knowledge intensive tasks but does not generalize well to reasoning-intensive tasks. A parallel area of research is improvements to prompt engineering and response decoding. A new method dubbed Chain of Thought concatenates hand-annotated example responses with reasoning to the query to create the prompt [3]. The responses from the LLM mimic the examples and contain a 'reasoning path' followed by the answer. Another method developed a complementary decoding approach, named Self-Consistency, with the idea being that marginalizing over several reasoning paths provides the best possible response [4]. However, these approaches rely heavily on human annotations and the same static examples may not be relevant to different queries.

Method: Combinatorial Reasoning (CR) supplements these approaches, dynamically creating the best reasoning path to answer a query without the need for human annotations. Inspired by Self-Consistency, we postulate that creating "Chain-of-Thought" style reasoning paths can be modeled as a discrete optimization problem. We first sample the LLM for reasons relevant to the query and then translate our sampling outcomes into a quadratic unconstrained binary optimization (QUBO) problem. Our optimization problem will consist of a set of linear terms and quadratic terms. Our linear terms, which represent the reasons being picked as part of the context, scale with how frequently reasons occur in our samples. Quadratic terms, which are terms associated with two reasons, scale based on how likely we are to observe two reasons occurring together in as sample. By selecting the optimal reasons from solving the QUBO, we create a prompt containing a relevant reasoning path for the original query. The current CR workflow, represented in figure 2, can be readily combined with retrieval based methods for the sampling and cutting-edge prompt engineering techniques for decoding.

Technique	LLM	Accuracy (%)
Zero-Shot	gpt-3.5-turbo	28.0
Chain of Thought	gpt-3.5-turbo	40.8
Self-Consistency	gpt-3.5-turbo	48.9
Combinatorial Reasoning	gpt-3.5-turbo	50.2

Figure 1: Accuracies of the methods on logical deduction 7 objects task. We recreated CoT and SC using the techniques described in references [3] & [4]. The tests were ran on a smaller subset of the whole task.

This Combinatorial Reasoning method developed by Icosa involves many hyperparameters. One such example is relative sensitivity to the linear and quadratic terms. Hyperparameter optimization is required to convert the best solution found through discrete optimization into the best possible response from an LLM. Such a hyperparameter optimization is provided as part of Icosa's commercial product.

The total time to get an answer is the following:

$$T_{total} = T_{sampling} + T_{optimization} + T_{prompting}$$

T_{total} = total time to solution
T_{sample} = time to generate multiple reasoning samples
$T_{optimization}$ = time to select optimal reasons
$T_{prompting}$ = time to execute CR prompt

When performing question and answer tasks over chat, users typically would like to see a response in less than 5 seconds. For the sampling of reasons, the LLM can be called in parallel, therefore T_{sample} is roughly the time to make a single LLM call ($\sim$ 1 second). The final prompting stage also involves making a single LLM call, so $T_{prompting} \sim 1$ second. Thus having a short $T_{optimization}$ in the 3 second range is important to create quick responses.

* mert@icosacomputing.com

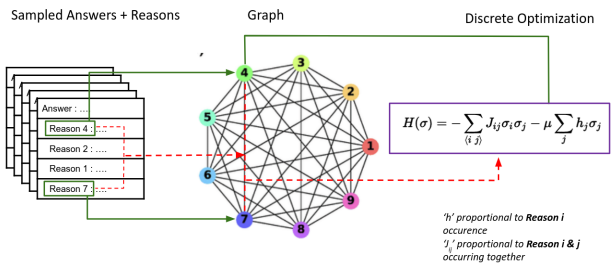

Figure 2: CR Workflow

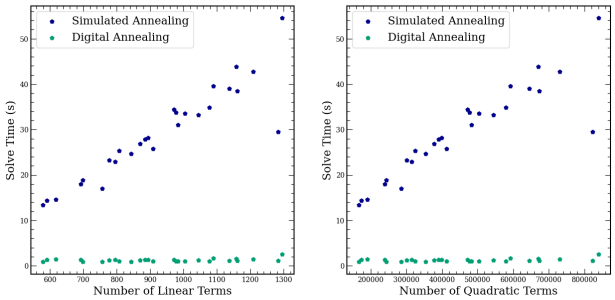

Figure 3: The solving times from Digital Annealing compared to those of simulated annealing. Digital Annealer's advantage grows as problem size increases.

Digital Annealer: Fujitsu's Digital Annealer is a special accelerator that speeds up combinatorial optimization, integrated into an operational setting with conventional hardware in a hybrid environment. It provides up to 10,000 times faster performance than industry standard compute systems running with commercial servers [5], and it already supports an 100,000-bit, fully-connected architecture. This ground-breaking offering is inspired by the key characteristics of quantum computing: superposition, quantum tunneling and entanglement, enabling the Digital Annealer to evaluate multiple potential options simultaneously – and to deliver lightning-fast insights.

By formulating improvements to LLMs as a discrete optimization problem, combinatorial reasoning can leverage the Digital Annealer to provide higher quality responses and avoid timing and selection problems common to classical methods. Solving large QUBO problems – an essential part of Combinatorial Reasoning – takes minutes to hours using classical methods. The current use-cases for LLMs depend on their quick response times, and thus long inference times is a potential roadblock for Combinatorial Reasoning. Furthermore, such classical techniques can find suboptimal solutions if the optimization landscape is glassy [6]. The Digital Annealer circumvents these issues, solving the QUBO problems near instantaneously while also providing higher quality solutions, leading to higher quality LLM responses.

Results: To benchmark Combinatorial Reasoning as well as Fujitsu's Digital Annealer, we looked at fixed answer text generation tasks that require reasoning. An industry standard collection of datasets for this task is Big Bench Hard [7], and we selected Logical Deduction (7 objects) from this as the dataset of interest. Logical Deduction tasks provide information about the ordering of a collection of objects and then query about the location of a specific object. CR has demonstrated better accuracy than methods such as Chain of Thought and Self-Consistency on this task (See Figures 1). As a dynamic method, CR creates a specific prompt with the best reasoning path tailored to the question. It overcomes the limitations of methods requiring human-annotated static examples, and thereby improves the accuracy on reasoning intensive tasks.

The tuned CR model used for this benchmark requires solving QUBO problems with (on average) 900 linear fully connected variables. While classical solvers are able to tackle problems of this size, they require long wait times before optimal solutions are reached. We note that for the classical solver, we used D-Wave Systems' open-source implementation of simulated annealing with 1000 sweeps and 100 reads. We selected this configuration as a lower number of sweeps and reads found poorer solutions and impacted performance. Figures 3 & 4 demonstrates the 10-50x speed up that the Digital Annealer provides while solving the same QUBO problems. We note that such fast solving time is crucial in integrating combinatorial reasoning into current language model use cases. Furthermore, there exist certain problems where — due to combination of problem sizes and difficult optimization landscapes — the Digital Annealer finds higher quality solutions in terms of the energy. We observed this phenomena for another reasoning dataset from Big Bench Hard: *formal fallacies*. We note that this was obtained with a different CR model with no hyperparameter tuning. With Icosa's proprietary hyperparameter tuning, the better solutions corresponding to lower energies can be converted into superior LLM responses.

Conclusions: Our results show that Combinatorial Reasoning was able to achieve slightly better results than CoT and Self-Consistency with no need for hand annotations. Using Fujitsu's digital annealer, we were able to get the total solution time that is in the 5 second range.

We note that the speedup we observed is not the same as the time we spent waiting for a solution. Even though we have not found a conclusive explanation for this phenomena, we believe that there are three possible explanations. Each time we use the Digital Annealer to solve an optimization problem, the environment may be going through a setup process which results in additional time being spent. There is also the concern that, due

* mert@icosacomputing.com

Number of Problems	Avg. # of Linear Terms	Avg. # of Quadratic Terms	Avg. Time Difference (s)
2	585.0	170838.0	-12.7
3	670.0	224794.0	-15.9
3	776.0	300827.0	-19.9
5	859.8	369696.0	-25.4
4	960.8	461495.3	-30.2
4	1053.8	555199.5	-34.0
3	1152.0	663033.0	-39.2
3	1263.0	797694.0	-40.5

Figure 4: The time-improvement of Digital Annealing over classical methods. The time difference column is time taken by classical method subtracted from time taken by Digital Annealer.

to the size of the problems we are tackling, a substantial amount of time may be spent on transferring the necessary data from our environment to the Digital Annealer's environment. Finally, it may be the case that, due to our shared access to the Digital Annealer, some of the excess time spent on waiting for a solution may be caused by having to wait for access while the Digital Annealer is being used.

However, despite the minor issues, the performance of the Digital Annealer is extremely promising. The speedups which we have observed will be crucial for the commercial applications of our combinatorial reasoning methods. Users cannot be expected to wait for up to almost a minute for classical methods to provide a solution, when the Digital Annealer can find the same solution in a fraction of the time. Furthermore, our recent discovery of problems which classical solvers greatly struggle with provide an entirely new area in which the Digital Annealer will be necessary. We believe that these problems will demonstrate that quantum-inspired methods such as the Digital Annealer will provide tangible improvements to generative language models.

Acknowledgements: We would like to thank the David Fellah and Calvin Barnum for helpful discussions. We also acknowledge the generous access provided by Fujitsu to their Digital Annealer. We thank Michiyuki Tanaka, Yasuyuki Tanaka, Hirofumi Ukita, and Atsushi Kasugai for their support and help with this work.

References

[1] Huang et al. "A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions". In: (2023). DOI: https://doi.org/10.48550/arXiv.2311.05232.

[2] Lewis et al. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks". In: (2020). DOI: https://doi.org/10.48550/arXiv.2005.11401.

[3] Wei et al. "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". In: (2022). DOI: https://doi.org/10.48550/arXiv.2201.11903.

[4] Wang et al. "Self-Consistency Improves Chain of Thought Reasoning in Language Models". In: (2022). DOI: https://doi.org/10.48550/arXiv.2203.11171.

[5] Xeon multi-processor system. The performance comparison was conducted by evaluating the quadratic assignment problem (QAP) on the Digital Annealer against a general purpose multi-core.

[6] Glassy refers to a cost landscape with several local minima and different configurations lead to similar low energies.

[7] Suzgun et al. "Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solv e Them". In: (2022). DOI: https://doi.org/10.48550/arXiv.2210.09261.

* mert@icosacomputing.com